Section 1: Cover Sheet

Final Report

Period of Performance: 12/1/07 – 11/30/10

Principal Investigator: Suvrajeet Sen
Affiliation: The Ohio State University
Address: Industrial and Systems Engineering
Baker Systems Engineering Bldg
1971 Neil Avenue, Columbus, OH 43210

Award Number: FA9550-08-1-0154

Project Title: *Models and Algorithms involving very large scale stochastic mixed-integer programs (SMIP)*

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
28 FEB 2011	**Final**	**01-12-2007 to 30-11-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Models and Algorithms involving very large scale stochastic mixed-integer programs	**FA9550-08-1-0154**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Sen Suvrajeet**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Ohio State University,Enarson Hall 154 W 12th Avenue, Columbus,OH,43210**	8. PERFORMING ORGANIZATION REPORT NUMBER **; AFRL-OSR-VA-TR-2011-0224**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **AFOSR, 875 North Randolph Street, Suite 325, Arlington, VA, 22203**	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **AFRL-OSR-VA-TR-2011-0224**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Stochastic Mixed Integer Programs (SMIP) are recognized as one of the most formidable classes of mathematical programming problems. Not only are there significant challenges due to potentially large number of scenarios, but, SMIP with integers in the second stage give rise to a non-convex and discontinuous recourse function that may be difficult to optimize. As a result of this project, there have been significant advances in the design of algorithms for solving SMIP problems. Thanks to this project, we are able to report on solution to SMIP problems with over a million binary variables in less than 3 hours of computing on a desk-top machine! These problems arise in a variety of Air Force applications, such as adaptive command and control (AC2) under uncertainty. For instance, consider a situation in which assignments of pilots/aircrafts to targets may be contingent of sensor data revealed during the mission. In such situations, a preliminary set of assignments are made, recognizing that these will be revised once more reliable observations (regarding targets) are available. While such adaptive methods can enhance the effectiveness of C2, uncertainties can vastly enlarge the set of choices, and new algorithmic tools are necessary. Our algorithms solve such problems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**17**	

Section 2: Objectives

The main purpose of this research project is to study new algorithms for Very Large-scale Stochastic Mixed-Integer Programming (VL-SMIP) problems. These problems arise in a variety of defense applications, such as adaptive command and control (AC^2) under uncertainty. For instance, consider a situation in which assignments of pilots/aircrafts to targets may be contingent of sensor data revealed during the mission. In such situations, a preliminary set of assignments are made, recognizing that these will be revised once more reliable observations (regarding targets) are available. While such adaptive methods can enhance the effectiveness of C^2, uncertainties can vastly enlarge the set of choices, and new algorithmic tools are necessary. Another potential use of VL-SMIP arises in selecting base locations from which to respond so that damage to valuable assets can be minimized during simultaneous attacks on several assets.[1]

The basic research that allows us to address VL-SMIP problems are as follows.

- Thrust A: Algorithms for Very Large-scale Stochastic Programming. Our approach to stochastic programming is based on decomposition-coordination algorithms which work on smaller pieces of a VL-SMIP, and coordinate these pieces by using novel convexification methods based on disjunctive programming ([A.1], [A.2][2]). For VL-SMIP, we have also designed specialized cut generation methods which are Stochastic Linear Programming (SLP) extensions of the traditional Cut Generation LP (CGLP) used for deterministic MIP problems. In addition, to VL-SMIP, we have also devised methods that accelerate cut generation for SLP problems by using re-sampling (A.).
- Thrust B: Algorithms for Deterministic MIP. Although the main focus of our research is on VL-SMIP, algorithms for the latter require advances for deterministic MIP algorithms which can be efficiently integrated with Stochastic Programming (SP), leading to algorithms for the VL-SMIP. Indeed, one of the more important results of this project is the discovery of a pure cutting plane algorithm that can solve, in finite time, a general deterministic MIP. Prior to our result, pure cutting plane algorithms were only available for binary MIP, and the question of finiteness had been open since the first MIP cutting planes due to Gomory in the 1960's. As expected, this discovery is being leveraged for the solution of VL-SMIP.

In this report, we provide overviews of each of these thrusts, with further details available in the papers that have resulted from the grant.

[1] *Research that is of particular relevance to the Air Force is highlighted in this manner*

[2] References to work in thrust A are referred to as [A.x], while those for thrust B are [B.x]. Citations other than those produced as a result of this project are listed as [C.x].

Section 3: Accomplishments / New Findings: Research Highlights and Relevance to the Air Force Mission

This section presents the principal research accomplishments of this project. It is organized into four main sub-sections, each representing summaries of two important findings for each thrust area. These new nuggets of knowledge cover a wide array of discrete optimization research, ranging from new theory, algorithms, and computational experiments. It should be noted that the success for VL-SMIP is predicated on advances in both MIP as well as SLP. Consequently, we present the results in that order. In each section we first describe the *challenge*, and then provide a *synopsis of the research* approach.

3.1 Algorithms for Very Large-scale Stochastic Mixed-Integer Programming (Thrust A)

Stochastic Mixed Integer Programs (SMIP) are recognized as one of the most formidable classes of mathematical programming problems. Not only are there significant challenges due to potentially large number of scenarios, but, SMIP with integers in the second stage give rise to a non-convex and discontinuous recourse function that may be difficult to optimize. [C.1] provides an illustration of how the presence of integer variables in the second stage leads to extremely complicated (non-convex and discontinous) recourse functions. Over the past few years, there have been significant advances in the design of algorithms for solving SMIP problems (see [C.8] and [A.5] for surveys). However, computational implementations, and results for large scale problems have been slow in coming. Thanks to this project, we are able to report on solution to SMIP problems with over a million binary variables in less than 3 hours of computing on a desk-top machine.

VL-SMIP Challenge

In order to maintain simplicity in this presentation, we assume that the second-stage problem satisfies the complete recourse property. Assuming that the random variable modeling uncertainty is discrete, with finite support $\Omega = \{\omega^1, \ldots, \omega^N\}$, a two-stage SMIP may be stated as

$$\text{(3.1.1)} \quad \text{Min } c^\top x + \sum_{\omega\in\Omega} p(\omega) g(\omega) y(\omega)$$
$$\text{(3.1.2)} \quad \text{s.t. } Ax \geq b$$
$$\text{(3.1.3)} \quad T(\omega)x + Wy(\omega) \geq r(\omega) \quad \forall \omega \in \Omega$$
$$\text{(3.1.4)} \quad (x, \{y(\omega)\}_{\omega\in\Omega}) \geq 0, x_j \text{ integer}, j \in J_1, y_j(\omega) \text{ integer}, j \in J_2.$$

Despite the fact that there are several assumptions underlying problem (3.1.1)-(3.1.4), it is somewhat general from the IP point of view since both the first and second stage variables allow general mixed-integer problems.

Suppose now that we wish to apply a resource directive decomposition method (i.e. Benders' decomposition). At iteration k of such a method, we solve one second-stage

subproblem for each outcome ω, and assuming that there is an appropriate solution method for the second-stage, then we can obtain a non-decreasing price function $h_k(\omega;\cdot)$ for each outcome $\omega \in \Omega$ [C.3]. Consequently, one obtains an inequality of the form

(3.1.5) $\quad \eta \geq \bar{H}_k(x) = \sum_{\omega\in\Omega} p(\omega) h_k\big(\omega; (r(\omega) - T(\omega)x)\big).$

Hence, as iterations proceed, one obtains a sequence of relaxed master programs of the following form.

(3.1.6) $\quad \text{Min } c^\top x + \eta$
(3.1.7) $\quad \text{s.t. } Ax \geq b$
(3.1.8) $\quad -\bar{H}_t(x) + \eta \geq 0, \ \forall\, 1 \leq t \leq k$
(3.1.9) $\quad x \geq 0, x_j \text{ integer}, j \in J_1.$

Because the approximation in (3.1.5) (and consequently (3.1.8)) is non-convex, the master problem in (3.1.6)-(3.1.9) is a non-convex mixed-integer program, and as indicated in [C.1], this approach is not scalable without further assumptions.

Synopsis of the Research

Our research is motivated by the observation that stochastic programming problems have a special (decomposable) structure, and in order to solve them in an efficient way, one needs to take advantage of their special structure. Our approaches use disjunctive programming ideas to approximate the feasible region by a convex set, and similarly, the expected recourse function is also approximated using disjunctive programming in a decomposable manner. Before summarizing the benefits of such an approach, we describe one of the concepts that is central to our research: disjunctive decomposition with a branch-and-cut framework for expected recourse approximations, referred to as D^2-BAC. Because D^2-BAC uses a Branch-and-Bound (B&B) search in the second-stage, it allows an "approximate solve" of the second-stage problem by truncating the number of nodes explored in the B&B search tree. Moreover, note that a B&B search tree that is obtained for one scenario (ω) can be used to approximate the second-stage MILP associated with another scenario.

Let $Q(\omega)$ denote the set of nodes of the B&B tree that have been explored for the subproblem associated with scenario . For any node $q \in Q(\omega)$, let $z_{q\ell}(\omega)$ and $z_{qu}(\omega)$ denote vectors whose elements are used to define lower and upper bounds, respectively, on the second-stage (integer) variables. In some cases, an element of $z_{qu}(\omega)$ may be $+\infty$, and in this case, the associated constraint may be ignored, implying that the corresponding dual multiplier is fixed at 0.

In any event, suppose that for a given $x \in X$, we wish to construct a lower bounding approximation of the value function of the second-stage problem. The LP relaxation for node q may be written as

(3.1.10) $h_q(\omega; x) = \min g^\top y$

(3.1.11) $Wy \geq r(\omega) - T(\omega)x$

(3.1.12) $y \geq z_{q\ell}(\omega), -y \geq -z_{qu}(\omega)$

The above LP can be used to derive a lower bounding function for node q, that is, we obtain an inequality of the form

$$h_q(\omega; x) \geq \alpha_q(\omega) + \beta_q(\omega)^\top x$$

Let $\mathcal{E}_q(\omega)$ denote the epigraph of the above affine approximation. Since the optimal value of the second-stage problem for a particular outcome (ω) is at least as large as the minimal lower bound, we obtain a lower bound on the subproblem value function for an outcome ω by defining a function

(3.1.13) $h_q(\omega; x) \geq \text{Min}_{q \in Q(\omega)} \alpha_q(\omega) + \beta_q(\omega)^\top x.$

Since right hand side in (3.1.13) is a non-convex function, using it directly in the first stage would result in a Benders' master program which is non-convex, leading to a problem that is not any easier than (3.1.5). Nevertheless we can convexify (3.1.13), thus leading to a more manageable master program. This is done by using disjunctive programming to derive a facet of the set

$$\mathcal{E}(\omega) := \cup_{q \in Q(\omega)} \text{clconv}\, \mathcal{E}_q(\omega).$$

For each $\omega \in \Omega$, a facet of $\text{clconv}\, \mathcal{E}(\omega)$ provides an affine lower bounding approximation of (3.1.13). Taking expectations then yields an affine approximation which replaces (3.1.8) in the master program, and now, the objective function of the master program is restored to being a piecewise linear and convex function, leading to a much more tractable approximation.

Table 1 summarizes computational results that have appeared in [A.9]. These instances are stochastic server location problems (SSLP) in which servers have to be located to meet demand which may or may not be realized in the future [C.6]. If demand is realized, then servers have to be allocated in such a way that capacity restrictions are satisfied and certain operating rules (e.g. servers can be only a specified distance from demand) must be satisfied, while meeting as much demand as possible. As shown in [A.4] SSLP problems which are essentially unsolvable without decomposition. (Here unsolvable refers to the inability to obtain an optimal solution within 10,000 seconds of computing on a Sun Fire 400 workstation.) However, Table 1 demonstrated that these are being solved to optimality using various versions of Disjunctive Decomposition (D^2). The earliest version, referred to as D^2 in Table 1, uses only set convexification, whereas the D^2-BAC algorithm uses both value function and set convexification [C.11]. The first two columns represent ideas that were implemented in [A.4], and the column labeled D^2-BAC+SLP refers to a new cut generation process which uses stochastic linear programming, and a corresponding specialized SLP decomposition to solve the cut formation LP in [A.9]. The improvements are clearly remarkable, considering that most of these instances are unsolvable using state-of-the-art deterministic MILP solvers [A.4].

The instances in Table 1 were first reported in [C.6], and subsequently used as a test-bed for evaluating advances for this genre of decomposition algorithms. Except for the first

two instances listed in the following table, we were unable to obtain optimal solutions using state-of-the-art commercial software. To appreciate the significance of some of the instances, note that if one were to use a deterministic equivalent formulation of the instance denoted by 10.50.2000, then the resulting problem would have at least one million binary variables (10×50×2000). In the decomposition framework however, the number of first-stage variables is 10 and the number of second-stage variables is 500 (per scenario). Because of decomposition, the 2000 sub-problems are solved independently, and as a result we are able to solve the high dimensional deterministic equivalent by solving a collection of lower dimensional problems. This is why decomposition is a winning strategy.

Table 1: Performance of Various Decomposition Algorithms for VL-SMIP

Instance	Binary Vars.	D2 (secs)	D2-BAC (secs)	D2-BAC+SLP (secs)
5.25.50	6255	1.64	0.70	0.36
5.25.100	12505	2.15	1.73	0.89
5.50.100	25005	7.10	3.70	1.56
5.50.500	125,005	34.50	23.05	12.36
5.50.1000	250,005	140.47	64.17	22.77
5.50.2000	500,005	603.37	274.40	42.74
10.50.50	25,010	295.95	373.98	262.13
10.50.100	50,010	396.76	452.31	486.99
10.50.500	250,010	1902.2	2772.22	1313.38
10.50.1000	**500,010**	**5410.1**	**5677.80**	**2139.47**
10.50.2000	**1,000,010**	**9055.29**	**> 10800**	**3916.47**
15.45.5	3,390	110.34	232.30	211.79
15.45.10	6,780	1494.89	222.41	153.41
15.45.15	10,080	7210.63	1988.26	803.56

As a result of such speed-ups, a collaborative DARPA project between AT&T and Telecordia is planning to implement these tools for a new generation of design tools for communications networks.

3.2 Convexification of Mixed-Integer Programming Problems (Thrust B)

SMIP algorithms presented in section 3.1 rely heavily on convexification of MIP problems. Algorithms for both master and subproblems depend on this operation. Disjunctive and lift-and-project cuts [C.2, C.3], semidefinite relaxations and reformulation-linearization technique (RLT) [C.12] have provided alternative approaches to generate cutting planes that define the convex hull of feasible points of a binary MILP. The case for MILP with general integers has not been as well understood, even when the integer variables are bounded. While pure integer programming can be shown to have finite representation using Gomory cuts, the same is not true for general MILP [C.5], and prior to the current project, this result was not available for disjunctive programming methods.

MIP Challenge

While disjunctive cuts are natural to use for general MIP problems, a cutting plane procedure using disjunctive cuts has not been proved to be finitely convergent. Indeed, the facial disjunctive property [C.2] was deemed critical for finite convergence and this property holds for binary MIP, but not for general MIP. Unfortunately, as shown in Figure 2 of [C.10], the absence of the facial disjunctive property could lead to infinitely many iterations. Subsequently, [C.7] provided the proof that in the absence of the facial disjunctive property, a one-variable-at-a-time method for convexifying disjunctive sets leads to an infinite convergent process that ultimately does provide the convex hull of feasible points. Of course, one could write a binary expansion of each general integer variable and the resulting formulation is a mixed 0-1 program which can be sequentially convexified. However, this is well known to be very inefficient because of a large number of variables, and the loss of any structure of the model.

We address the following questions:

- Is it possible to use the disjunctive programming methodology to describe the convex hull of MILP solutions in finitely many steps without introducing binary variables?
- Is there a constructive methodology to obtain an optimal solution to a general MIP using a disjunctive programming characterization of its convex hull?
- If we are restricted to introduce only one cutting plane in any iteration, is there a finitely convergent disjunctive programming algorithm that solves a general MIP?

Synopsis of the Research

In this synopsis, we restrict our presentation to only the last bullet mentioned above. This is in fact the answer to a question that has been open for decades [C.5]. For details regarding the other bullets, see [B.1] where the concept of cutting plane tree algorithm was introduced.

In the cutting plane tree $\mathcal{T}$, there is a single root node o. For each node $\sigma \in \mathcal{T}$, an integer m_σ keeps track of the cutting planes that will be used to generate a disjunctive cut when this node is revisited, an integer $v_\sigma \in \{1, 2, \dots, n_1\}$ stores the index of the integer variable that is split, an integer q_σ stores the (lower) level of the splitting. Let l_σ, r_σ and p_σ denote links to the left child, right child and parent nodes of node σ, respectively. Let $\mathcal{S}(\sigma)$ be all nodes on the sub-tree rooted at node σ (not including node σ and the leaf nodes).
Let $\mathcal{N}(\sigma)$ be the collection of the nodes on the path from the root node to node σ (not including the root node), let $\mathcal{N}^-(\sigma)$ be the collection of nodes in $\mathcal{N}(\sigma)$ that were formed as the left child node of its parent, and let $\mathcal{N}^+(\sigma)$ be the collection of nodes in $\mathcal{N}(\sigma)$ that were formed as the right child node of its parent. Given $\sigma \in \mathcal{T}$ define

$$\mathcal{C}_\sigma = \{ x \,|x_j \in [0, u_j], x_{v_{p_s}} \leq q_{p_s}, \forall\, s \in \mathcal{N}^-(\sigma), x_{v_{p_s}} \geq q_{p_s} + 1, \forall\, s \in \mathcal{N}^+(\sigma)\}.$$

We let m_σ store an iteration index, which gives the set X_{m_σ} to be used in the cut generation LP (CGLP) (Balas 1979, Sherali and Shetty 80). The set X_{m_σ} corresponds to the intersection of the linear relaxation (X_L) together with the first $m_\sigma - 1$ cuts. If

$X_{m_\sigma} \cap \mathcal{C}_{l_\sigma} = \emptyset$ ($X_{m_\sigma} \cap \mathcal{C}_{r_\sigma} = \emptyset$), we say that the left (right) child node of σ is "fathomed'", i.e., $l_\sigma = null$ ($r_\sigma = null$). Let $\mathcal{L}_{k+1}$ denote the collection of all leaf nodes of the cutting plane tree at the end of iteration *k*. Then, the set of mixed-integer feasible points belong to the union of $\mathcal{C}_\sigma$ for $\sigma \in \mathcal{L}_{k+1}$.

At iteration *k*, if the current extreme point solution to $\min_{x \in X_k} c^\top x$, given by x^k is integral, then we have found the optimal solution to the MIP. Otherwise, we search the cutting plane tree, to find the last node σ on the path from the root node such that $x^k \in \mathcal{C}_\sigma$. There are two cases: Case (1) σ is a leaf node ($\sigma \in \mathcal{L}_{k+1}$), Case (2) σ is not a leaf node ($\sigma \notin \mathcal{L}\_k, x^k \in \mathcal{C}_\sigma$, $x^k \notin \mathcal{C}_{l_\sigma}$ and $x^k \notin \mathcal{C}_{r_\sigma}$). In Case (1), we choose a fractional variable $x_j, j = 1, \dots, n_1$ with the smallest index, and let the split variable be $v_\sigma = j$. We create two new nodes: left (l_σ) and right (r_σ) children of σ at the split level $q_\sigma = \lfloor x_j \rfloor$. We let $\mathcal{C}_{l_\sigma} = \{x \in \mathcal{C}_\sigma \mid x_j \leq \lfloor x_j^k \rfloor\}$ and $\mathcal{C}_{r_\sigma} = \{x \in \mathcal{C}_\sigma \mid x_j \geq \lceil x_j^k \rceil\}$. In this case, we also let $m_\sigma = k$, as this is the first time the tree search for a fractional solution stops at node σ. In Case (2), the cutting plane tree and m_σ are unchanged. However, in this case, we update $m_t = k$ for all successors of σ, $t \in \mathcal{S}(\sigma)$. We generate a valid inequality for the set $\text{clconv}\{\cup_{t \in \mathcal{L}_{k+1}} (X_{m_\sigma} \cap \mathcal{C}_t)\}$ that cuts off x^k (using an extreme point of the Cut Generation LP). The new inequality is included along with those defining X_k, and the resulting set is denoted X_{k+1}. This process continues until one of the stopping criteria is satisfied.

Theorem: Assume that the set of feasible solutions of an MIP is non-empty and has bounded integer variables. Then the cutting plane tree algorithm converges to the optimal solution in finitely many iterations. (See Chen, Kucukyavuz and Sen 2009).

The above result settles a question that has been open since the inception of mixed-integer programming fifty years ago!

3.3 Computational Implications of the Cutting Plane Tree Algorithm (Thrust B)

As with deterministic MIP problems, cutting plane schemes are indispensable for stochastic MIP problems. Section 3.1 clearly demonstrates the power of combining cutting planes with decomposition. However, some major challenges remain before we can extend these ideas to general stochastic mixed-integer programs.

The Challenge

The CPT represents an adaptive sequence of disjunctions involving multiple variables, and is able to discover the convex hull (closure) of any instance of a bounded mixed-integer programs with general integer variables (MIP-G), without having to specify an a priori hierarchy. This characterization is a generalization of the sequential convexification process of [C.2] for MIP with binary variables (MIP-B). However, it is important to note that the same sequential process of convexification (one variable at a time) does not yield the convex hull of MILP-G in finitely many steps [C.7]. Prior to this [C.10] presented examples of non-convergence in which facet inequalities of two-term

(simple) disjunctions are derived to cut away the solution to the most recent LP relaxation. The convexification result presented in [B.1] (Section 3.2) certainly indicates that there may be hope in that direction, but is this realizable in a computationally viable algorithm? The challenge is to solve a battery of test problems using only simply cutting plane methods. Such a test, reported in [B.2] avoids the effects of other tools commonly used in MIP solvers, thus giving a better indicator of their strength.

Computational Experiments with CPT

In order to isolate and identify the potential of our scheme of generating multi-term disjunctions, we do not include other computational devices such as branch-and-bound, other classes of cutting planes, heuristics or preprocessing strategies. Thus, the CPT algorithm was tested as a pure cutting plane method. Moreover, two different types of cut generation LPs (CGLP) were devised: one was the weighted cut coefficients (WCC) formulation commonly used in lift-and-project implementations of the disjunctive cut principle, and the other is a new normalization scheme based on minimizing the 1-norm of the cut coefficients (M1NC). The instances used for this study were selected from the MIPLIB 2.0, 3.0 and 2003 libraries[3] that have total number of variables less than or equal to one thousand and rows less than or equal to one thousand. The computer code was implemented in C in Microsoft Visual Studio 2003. We conducted our experiments on Windows XP platform with Intel Q9450 Core 2 Quad processor, with 4 cores, 4 threads, running at 2.66 GHz speed with 4 GB of RAM. The linear programming solver is IBM ILOG CPLEX 12.2.

The computational results for each variant is shown in Tables 2 and 3 in the following pages. Table 2 reports the performance for Binary Mixed-Integer Programs, whereas, Table 3 reports the same for General Mixed-Integer Programs. The measure of performance that we are interested in is the gap closure (Gapcl) in these tables. While some of the harder problems show no gap closure (which is consistent with the other cutting plane schemes in the literature), the degree of gap closure obtained after an hour of computing exceeds 50% on average, in both tables. This is indeed remarkable because we have not used any branching, or multiple specialized cutting planes, as is commonly done by commercial software.

[3] The test instances are available at http://miplib.zib.de

Table 2: Performance of CPT on Standard *Binary* MIP Test Problems

	WCC					M1NC				
Instance	Time(s)	Cuts	T	N	Gapcl	Time(s)	Cuts	T	N	Gapcl
aflow30a	>3600	794	2	3	42.88%	>3600	1456	2	3	70.85%
danoint	>3600	398	2	3	2.64%	>3600	500	5	9	1.74%
dcmulti	>3600	975	2	3	99.51%	>3600	1390	4	8	100.00%
egout	6.6	173	2	3	100.00%	2.1	120	3	5	100.00%
enigma	>3600	1524	4	7	nogap	>3600	3324	9	18	nogap
fixnet6	>3600†	675	2	3	69.13%	>3600	1132	4	7	90.86%
glass4	21.8*	192	2	3	75.00%	>3600	12	4	7	25.00%
lseu	>3600	2068	2	3	47.61%	>3600	2983	3	5	56.94%
markshare1	>3600	399	11	21	0.00%	>3600	2358	25	49	0.00%
markshare2	>3600	185	8	15	0.00%	>3600	2475	20	39	0.00%
mas74	>3600	14	2	3	11.01%	>3600	0	1	1	0.00%
mas76	>3600	4	3	5	7.05%	>3600	0	1	1	0.00%
misc03	>3600	1862	3	5	55.95%	>3600	8755	2	3	57.26%
misc07	>3600	1412	3	5	12.03%	>3600	4859	4	7	12.06%
mod008	>3600	1413	3	5	24.95%	>3600	1710	2	3	31.62%
modglob	>3600	619	2	3	85.67%	>3600†	479	2	4	99.93%
opt1217	>3600	920	2	3	0.60%	>3600	1087	11	21	0.53%
p0033	>3600	280	3	5	72.92%	>3600	5203	6	11	99.35%
p0201	>3600	974	2	3	94.74%	>3600	2463	2	3	76.57%
p0282	>3600	1753	2	3	97.08%	>3600	2664	2	3	98.31%
p0548	>3600	1038	2	3	46.79%	>3600	1305	2	3	100.00%
pk1	>3600	969	7	13	0.00%	>3600	3373	6	11	0.00%
pp08a	>3600†	1169	2	3	98.13%	>3600	2355	2	3	99.50%
pp08aCUTS	>3600	519	2	3	89.33%	>3600	2178	2	3	98.88%
qiu	>3600	144	1	1	55.97%	>3600	876	2	3	81.38%
rgn	>3600	1144	3	5	46.48%	>3600	435	5	9	53.65%
set1ch	>3600†	1840	2	4	99.95%	2896.7	1675	5	14	100.00%
stein15	>3600	3914	2	3	24.19%	>3600	4406	2	3	38.50%
stein27	>3600	4010	2	3	12.93%	>3600	4344	2	3	24.63%
stein45	>3600	3051	2	3	0.000%	>3600	3507	2	3	0.000%
vpm1	>3600	1485	2	3	84.88%	206.6	1304	2	3	100.000%
vpm2	>3600	1445	2	3	79.48%	>3600	1723	2	3	88.14%
Average		1182.	2.8	4.7	49.58%		2201.	4.6	8.4	55.02%

Table 3: Performance of CPT on Standard *General* MIP Test Problems

	WCC					M1NC				
Instance	Time(s)	Cuts	T	N	Gapcl	Time(s)	Cuts	T	N	Gapcl
bell3a	>3600	74	5	9	70.74%	>3600	1320	9	17	74.62%
bell5	>3600†	503	2	4	94.28%	>3600	1089	4	6	97.19%
blend2	>3600†	440	3	5	64.95%	>3600	2551	2	3	41.46%
flugpl	>3600	1342	3	5	23.65%	>3600	8991	4	8	27.08%
gen	>3600†	698	3	6	91.94%	>3600	1881	2	3	96.08%
gt2	>3600	663	3	5	97.06%	>3600	1958	14	27	93.24%
noswot	>3600	315	7	13	0.00%	>3600	904	5	9	0.00%
rout	>3600	732	2	3	3.13%	>3600	2030	2	3	6.95%
timtab1	>3600	1132	2	3	37.50%	>3600	3018	3	5	44.02%
timtab2	>3600	1170	2	3	29.40%	>3600	2818	2	3	33.92%
Average		706	3.2	5.6	51.27%		2656.	4.7	8.4	51.46%

3.4 Extensions of Stochastic Decomposition (Thrust A)

The Challenges

Stochastic Decomposition is a sequential sampling algorithm which was designed to solve two-stage stochastic linear programming (SLP) problems. As far back as 1992, SD provided near-optimal solutions on very large SLP problems like SSN on work-station class computers of the time [C.9]. Nevertheless, it still had some limitations: a) it is unable to solve SMIP instances in which the second-stage has integer variables, and b) even with SLP problems, its optimality cuts require all previously observed outcomes, making cut generation more computationally intensive than may be necessary. The first limitation was overcome in the dissertation research of [A.9], funded as part of this project. This work not only allows us to handle SMIP instances within SD, but also allows the SMIP instance to take advantage of the sampling capability of SD, thus allowing the solution of instances with *continuous* distributions, and achieving convergence with probability one. The second challenge was addressed in [A.6] by introducing the notion of re-sampling during the cut generation process.

Synopsis of the Research

It turns out that the idea of re-sampling is important for SLP, but becomes much more critical for VL-SMIP. However, the specific goals and strategies for each class of problems differ in their details because the cut generation for SLP captures value (or expected recourse) function approximations, whereas, the cut generation in VL-SMIP refers to the convexification procedure using D^2. of the integer feasible points. Because the setup for SLP is more intuitive, we will discuss that first.

The cut generation process in SD requires that we approximate a subgradient for each outcome of the value function of the second stage LP. The re-sampling process simplifies this step by only choosing a fraction of the outcomes that have been generated until iteration *k*. One relatively straightforward way to include re-sampling within the cut generation process is to accept/reject an outcome within a sample, based on a Bernoulli random variable. Thus, if p is the acceptance (success) probability, then we generate a uniform random variate for each previously generated outcome, and use only those outcomes $(\omega^t, t < k)$ for which the random number is less than p. Clearly, as p increases, SD cuts tend to use more outcomes in the approximation. In the computational implementation of this process, we re-sample only after a certain minimum number of iterations have been completed because the re-sampling process can only benefit when the number of outcomes used for cut-generation is large.

We demonstrate the effectiveness of re-sampling by solving an instance referred to as 20Term which arose in Freight Scheduling (Infanger-1999). For this instance, we fix the maximum number of iterations at 800 for both versions of Regularized SD (with and without re-sampling) and perform 20 replicated runs. In the re-sampled version, we start the re-sampling process after 300 iterations and choose the acceptance probability p as 0.7. Finally, the LP solver in our computation uses the ILOG CPLEX callable library,

version 10.0, and all programs were compiled and run in a Unix environment running on a Sun workstation (Sun Fire V440).

Figure 1 reports the solution times for the two versions we are comparing. We record the CPU time (in seconds) every 100 iterations. As illustrated in Figure 1, there is no difference between the two versions for the first 300 iterations because re-sampling was started only after 300 iterations. However, after 300 iterations, the re-sampled version is faster as iterations proceed. Moreover, at iteration 800, the re-sampled version takes 62 seconds compared to 84.5 seconds for Regularized SD. Thus the re-sampled version results in a reduction of 26.7 % in computational time.

Figure 2 demonstrates the solution quality obtained by the two versions. As expected, there is no difference between the two versions SD for the first 300 iterations. At iterations 400, there is a jump in objective function value due to re-sampling. These values were obtained by running an out-of-sample evaluator that samples the objective function, given a first stage solution used in a particular iteration. It is interesting to observe that although the objective values obtained by the re-sampled version are not as good in the early iterations of re-sampling, the two versions begin to converge to the same value as iterations proceed. As a matter of fact, at iteration 800, one observes scant differences in objective function value between the two versions, with the original Regularized SD version yielding 254561.8310 and the sampled version providing a slightly higher value at 254572.1976.

Figure 1: Example savings in computational time using re-sampling

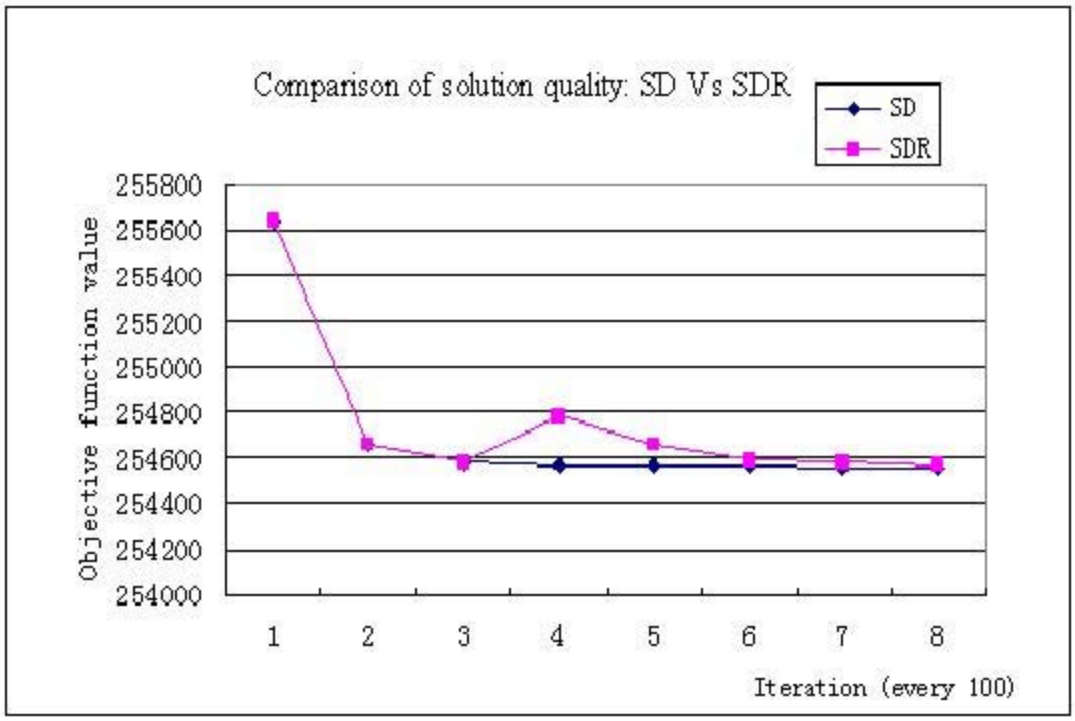

Figure 2: Example of negligible loss in optimality due to re-sampling

As for using SD for SMIP problems, asymptotic convergence was proved by Yuan (2010) for the modification in which D2 cuts are added in each iteration to all of the subproblems. While the inclusion of all outcomes in the optimality cut provides a stronger cut, this also increases the time it takes to generate such a cut. Instead, we use only a subset of the outcomes for optimality cut generation. This approach speeds up the computations as shown in Figure 3, where the largest instance of Table 1 was solved to optimality in 400 SD iterations, although our experiments ran the method to 1000 iterations, with no changes observed. The solution shown in Figure 3 are the same ones that were obtained using D^2-BAC+SLP.

Section 4: Personnel Supported

PI: Suvrajeet Sen

Graduate Students: Yang Yuan (graduated in 2010), Yunwei Qi (expected graduation in 2011) and Dinakar Gade (expected graduation 2012).

Post-doctoral Scholar: Shugang Kang.

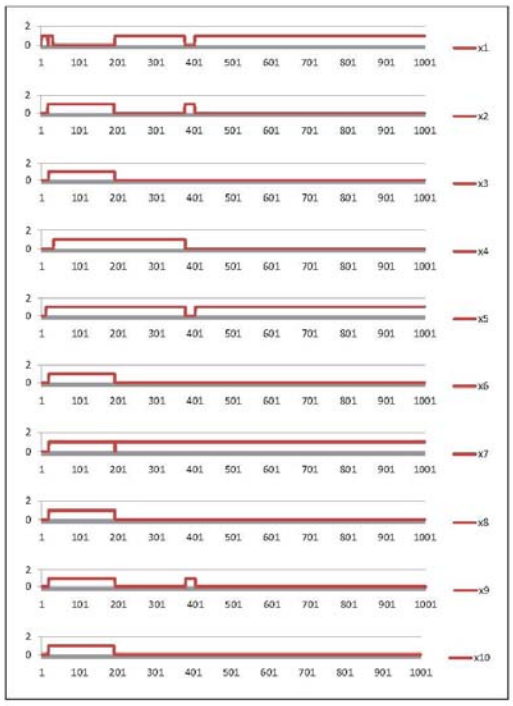

Figure 3: Optimal solution for SSLP instance 10×50×2000 using SD for VL-SMIP

Section 5: Publications

All papers listed below can be obtained by sending e-mail to the PI at sen.22@osu.edu. For those papers that are not covered in the synopses in the body, summaries of how they are relevant to the project, and important to the Air Force are provided below.

Thrust A: Algorithms for Very Large Scale Stochastic Programming

[A.1] T. Genc and S. Sen 2008, "An analysis of capacity and price trajectories for the Ontario Electricity Market using dynamic Nash Equilibrium under uncertainty," *Energy Economics*, 30, pp. 173-19.

Dynamic Nash games arise in a variety of applications for the Air Force. The methodology of this paper allows us to handle games in which the future is uncertain. This particular paper shows the applicability of the methodology of this project with large scale data from the electricity generation sector.

[A.2] J.L. Higle, B. Rayco, and S. Sen 2009, "Stochastic Scenario Decomposition for Multi-stage Stochastic Programs," *IMA Journal of Management Mathematics,* pp 1-28.

This paper reports on a multi-stage approach to stochastic decomposition by using a dual version of the multi-stage stochastic programming problem. We show how the search can be made more efficient by using aggregation-disaggregation principles. This type of algorithm is useful when the control actions must evolve with observed data over multiple decision epochs. This paper is very central to the goals of this project, and the needs of adaptive command and control under uncertainty.

[A.3] K. Huang, S. Sen, and F. Szidarovszky, "Connections among Decision Field Theory models of cognition," submitted to *Journal of Mathematical Psychology*.

Psychologists have described human decision-making through a class of models that are classified as Decision Field Theory (DFT). Through experimental investigations, these mathematical models have been identified as being important for understanding pilot decision-making, especially for time-critical decisions. Such models are clearly of value to the Air Force. Our research has provided a unfied theory for a variety of DFT models.

[A.4] L. Ntaimo and S. Sen 2008, "A Comparative Study of Decomposition Algorithms for Stochastic Combinatorial Optimization," *Computational Optimization and Applications*, vol. 40, pp. 299-319.

Discussed in the synopsis

[A.5] S. Sen 2010, "Stochastic Integer Programming Algorithms: Beyond Benders' Decomposition," accepted for publication in *Wiley Encyclopedia on Operations Research and Management Science*.

Discussed in the synopsis

[A.6] S. Sen, Z. Zhou and K. Huang 2009, "Enhancements of Two-Stage Stochastic Decomposition, *Computers and Operations Research*, pp. 2434 – 2439. An updated version entitled "Stochastic Decomposition and Extensions," is to appear in "Stochastic Programming: The State of the Art," in honor of George Dantzig, edited by G. Infanger.

Discussed in the synopsis

[A.7] S. Sen and Z. Zhou, "Multi-stage Stochastic Decomposition" submitted to *SIAM Journal on Optimization* (currently under revision).

This type of algorithm is useful when the control actions must evolve with observed data over multiple decision epochs. Clearly this paper is completely in line with the goals of this project, and the needs of the Air Force.

[A.8] Y. Yuan 2010, "Algorithmic Advances in Stochastic Combinatorial Optimization and Applications, Ph.D. dissertation, ISE Department Ohio State University, Columbus, OH.

Discussed in the synopsis

[A.9] Y. Yuan and S. Sen 2009, "Enhanced cut generation methods for decomposition-based branch-and-cut algorithms for two-stage stochastic mixed-integer programs," *INFORMS Journal on Computing*, pp. 480 – 487.

Discussed in the synopsis

Thrust B: Algorithms for Deterministic Mixed-Integer Programming

[B.1] B. Chen, S. Küçükyavuz, S. Sen, "Finite Disjunctive Programming Characterizations for General Mixed-Integer Linear Programs," accepted in *Operations Research*.

Discussed in the synopsis

[B.2] B. Chen, S. Küçükyavuz, S. Sen, "A Computational Study of the Cutting Plane Tree Algorithm for General Mixed-Integer Linear Programs," submitted to *Operations Research Letters*.

Discussed in the synopsis

Citations to the Literature

[C.1] S. Ahmed, M. Tawarmalani, N.V. Sahinidis. 2004. A finte branch and bound algorithm for two-stage stochastic integer programs, *Mathematical Programming*, **100**, 355-377.

[C.2] E. Balas, 1979. Disjunctive programming. *Annals of Discrete Mathematics* **5** 3-51.

[C.3] E. Balas, S. Ceria, G. Cornu\'ejols 1993, A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs. *Mathematical Programming*. 58:295-324

[C.4] C.C. Caroe, J. Tind, 1998, J. L-Shaped Decomposition of Two-Stage Stochastic Programs with Integer Recourse. *Mathematical Programming.* **83**, 139-152.

[C.5] A. Martin 2006, "Large Scale Optimization," *Encyclopedia of Life Support Systems.*

[C.6] L. Ntaimo, S. Sen. 2005. The Million Variable "March" for Stochastic Combinatorial Optimization. *Journal of Global Optimization* **32**.

[C.7] J.H. Owen, S. Mehrotra. 2001, "A disjunctive cutting plane procedure for general mixed-integer linear programs," *Mathematical Programming*, **89**, 437--448

[C.8] S. Sen, 2005. Algorithms for Stochastic Mixed-Integer Programming Models. *Handbook of Discrete Optimization*, (K. Aardal, G.L. Nemhauser, and R. Weismantel eds.), North-Holland Publishing Co., pp. 515-558.

[C.9] S. Sen, R.D. Doverspike and S. Cosares 1994, "Network Planning with Random Demand," *Telecommunication Systems*, **3**, 11-30.

[C.10] S. Sen, H.D. Sherali 1985, "On the convergence of cutting plane algorithms for a class of nonconvex mathematical programs," *Mathematical Programming*, **31**, 42--56

[C.11] S. Sen, H.D. Sherali. 2006. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming, Mathematical Programming, **106**, 203-223.

[C.12] H.D. Sherali, W.P. Adams. 1990. A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems, *SIAM Journal on Discrete Mathematics*, **3**, pp. 411-430.